BAB 3

ANALISIS DAN DESAIN SISTEM

3.1 Analisis Masalah Umum

Shortest path merupakan suatu persoalan untuk mencari lintasan antara dua buah *node* pada *graph* berbobot yang memiliki gabungan nilai jumlah bobot pada *edge graph* yang dilalui dengan jumlah yang paling minimum atau dapat dinyatakan juga sebagai berikut :

Diberikan sebuah *graph* berbobot (dengan himpunan *node* V, himpunan *edge* E, dan fungsi bobot bernilai bilangan *real* yang dapat ditulis dengan f : E→ **R**), dan diberikan elemen v' dari V , sehingga dapat dicari sebuah lintasan P dari v ke setiap v dari V, sehingga:

$$\sum_{min} f(P)$$

adalah nilai minimal dari semua lintasan yang menghubungkan v ke v'. Dimana f(P) adalah fungsi bobot (panjang) sebuah lintasan P dan p adalah elemen dari lintasan P.

3.2 Analisis Algoritma A*

Algoritma A* merupakan algoritma pencari jalan terbaik dan merupakan gabungan dari algoritma Djikstra dan *Best First Search* (BFS). Ketiga algoritma ini menggunakan *graph* berbobot tidak berarah sebagai konsep dasar pencarian jejak. Algoritma Djikstra mampu memastikan bahwa jalan yang terpilih adalah jalan terpendek. Namun Djikstra merupakan algoritma yang lamban karena cara kerjanya

brute force. Algoritma *Best First Search* (BFS) mampu mencari jalan lebih cepat, namun dalam beberapa kasus, bukan merupakan jalan terpendek. Algoritma A* yang merupakan gabungan dari kedua algoritma tersebut, mampu mencari jalan terpendek seperti Djikstra dan hampir secepat *Best First Search* (BFS).

Rahasia dari algoritma A* adalah pemilihan simpul untuk pencarian jejak didasarkan pada dua hal, yaitu memilih simpul yang lebih dekat ke posisi awal (prinsip Algoritma Djikstra) dan lebih dekat ke posisi akhir (prinsip Algoritma *Best First Seacrh*). Sama seperti *Best First Search* (BFS), Algoritma A* juga menggunakan fungsi heuristik yang menentukan urutan titik mana yang akan dikunjungi terlebih dahulu. Fungsi heuristik ini sebenarnya menyimbolkan seberapa baik atau mungkin titik itu dikunjungi untuk mencapai titik tujuan. Fungsi ini terdiri dari penjumlahan dua fungsi, yaitu :

1. Fungsi biaya jalan, yang merupakan biaya dari titik awal sampai titik sekarang. Biasanya disebut dengan g(n).
2. Fungsi perkiraan jarak menuju titik tujuan dari titik sekarang. Biasanya disebut dengan h(n). sesuai dengan namanya, h(n) hanyalah fungsi perkiraan yang dapat diterima.

A* menyeimbangkan kedua nilai ini dalam mencari jalan dari *node* awal ke *node* tujuan. Dalam menentukan *node* yang akan dikembangkan, algoritma ini akan memilih *node* dengan nilai *f(n)* yang paling kecil

Dengan perhitungan biaya seperti ini, algoritma A* adalah *complete* dan optimal [19]. Hal ini berbeda dengan *Greedy best-first search* yang hanya memperhitungkan biaya perkiraan saja, yaitu f(n) = h(n), biaya sebenarnya (*actual cost*) tidah diperhitungkan. Dengan hanya menggunakan biaya perkiraan yang belum tentu kebenaranya, maka algoritma ini tidak optimal.

Sama halnya seperti algoritma *best first search*, algoritma A* juga menggunakan dua antrian, yaitu OPEN dan CLOSED. Terdapat tiga kondisi bagi setiap suksesor yang dibangkitkan, yaitu : sudah berada di OPEN, sudah berada di

CLOSED, dan tidak berada di OPEN maupun CLOSED. Pada ketiga kondisi tersebut diberikan penanganan yang berbeda-beda.

Node yang berada di OPEN merupakan verteks yang pernah dibangkitkan, dan nilai heuristiknya telah dihitung, tapi belum terpilih sebagai *node* terbaik (*bestnode*). *Node* yang berada di OPEN memiliki peluang untuk terpilih sebagai *node* terbaik (peluang masih terbuka). Sebuah *node* akan disimpan di CLOSED jika *node* tersebut pernah terpilih sebagai *bestnode*. Dengan kata lain CLOSED berisi *node* yang tidak mungkin terpilih sebagai simpul terbaik (peluang untuk terpilih sudah tertutup).

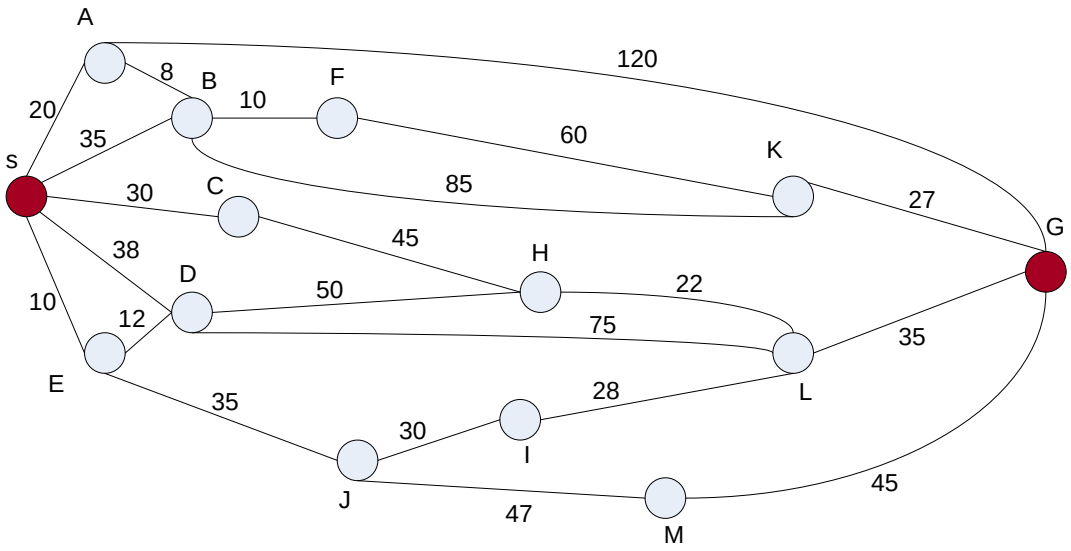
Jika suksesor sudah pernah berada di OPEN, maka dilakukan pengecekan apakah perlu pengubahan *parent* atau tidak tergantung pada nilai *g*-nya melalui *parent* lama atau *parent* baru. Jika melalui *parent* baru memberikan nilai *g* yang lebih kecil, maka dilakukan pengubahan *parent*. Jika pengubahan *parent* dilakukan, maka dilakukan pula perbaruan (*update*) nilai *g* dan *f* pada suksesor tersebut. Dengan pembaruan ini, suksesor tersebut memiliki kesempatan yang lebih besar untuk terpilih sebagai *node* terbaik (*bestnode*).

Jika suksesor sudah pernah berada di CLOSED, maka dilakukan pengecekan apakah perlu pengubahan *parent* atau tidak. Jika ya, maka pebaruan nilai g dan f pada suksesor tersebut serta pada semua “anak cucunya” yang sudah pernah berada di OPEN. Dengan perbaruan ini, maka semua anak cucunya tersebut memiliki kesempatan lebih besar untuk terpilih sebagai *node* terbaik (*bestnode*).

Jika suksesor tidak berada di OPEN maupun CLOSED, maka suksesor tersebut dimasukkan ke dalam OPEN. Tambahkan suksesor tersebut sebagai suksesornya *bestnode*. Hitung biaya suksesor tersebut dengan rumus :

$f(n)=g(n)+h(n)$

Untuk memperjelas pemahaman tentang algoritma A*, maka akan diberikan contoh masalah pada gambar dibawah ini :



Gambar 3.1 Contoh *Graph* untuk Pencarian Lintasan

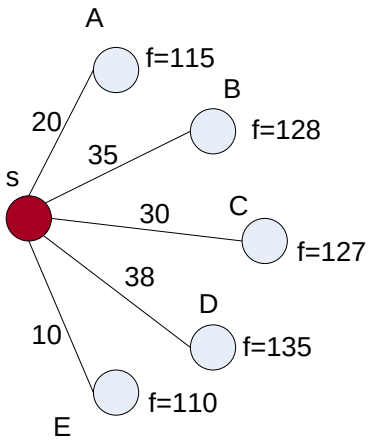
Masalah pencarian lintasan dalam suatu daerah yang direpresentasikan dalam suatu *graph* dua arah. Setiap *node* menyatakan suatu kota. Busur menyatakan jarak sebenarnya antara satu kota dengan kota lainnya dan h(n) menyatakan biaya perkiraan (jarak Euclid) dari *node* n menuju *node* G, maka akan dicari rute terpendek dari S menuju ke G.

Dengan menggunakan fungsi heuristik jarak Euclid, maka didapat h(n) masing-masing *node* sebagai berikut:

Tabel 3.1 Jarak Euclid masing-masing *node* pada Contoh Graph

S	A	B	C	D	E	F	G	H	I	J	K	L	M
110	95	93	97	107	100	85	0	45	45	78	25	27	40

Berikut adalah langkah-langkah untuk menyelesaikan permasalan lintasan terpendek dari *node* S ke *node* G dengan menggunakan Algoritma A*:



Gambar 3.2 Contoh *Graph* Langkah Pertama

Fungsi evaluasi:

f(A) = g(S) + g(S ke A) + h(A) = 0 + 20 + 95 = 115

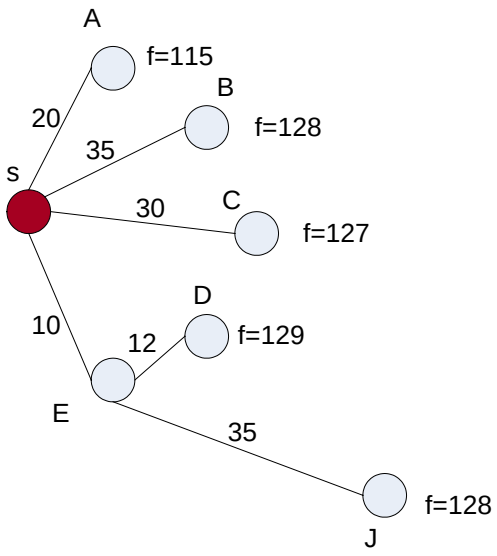
f(B) = g(S) + g(S ke B) + h(B) = 0 + 35 + 93 = 128

f(C) = g(S) + g(S ke C) + h(C) = 0 + 30 + 97 = 127

f(D) = g(S) + g(S ke D) + h(D) = 0 + 38 + 97 = 135

f(E) = g(S) + g(S ke E) + h(E) = 0 + 10 + 100 = 110

Pertama, karena di OPEN hanya terdapat satu verteks, yaitu S. Maka S terpilih sebagai *bestnode* dan dipindahkan ke CLOSED. Kemudian dibangkitkan semua suksesor S, yaitu : A, B, C, D dan E. Karena kelima suksesor tidak ada di OPEN maupun CLOSED, maka kelimanya dimasukkan ke OPEN. Langkah pertama ini menghasilkan OPEN=[A,B,C,D,E] dan CLOSED=[S].



Gambar 3.3 Contoh *Graph* Langkah Kedua

Fungsi evaluasi

f(A) = 115

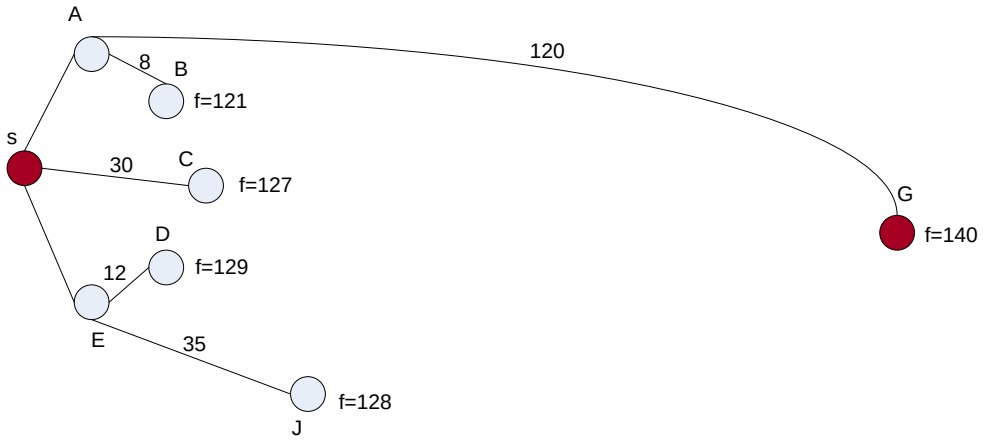
f(B) = 128

f(C) = 127

f(D) = g(E) + g(E ke D) + h(D) = 10 + 12 +107 =129

f(J) = g(E) + g(E ke J) + h(J) = 15 + 35 + 78 =128

Selanjutnya, E dengan biaya terkeci, yaitu 110, terpilih sebagai *bestnode* dan dipindahkan ke CLOSED. Lalu , semua suksesor E dibangkitkan, yaitu: D dan J. karena belum pernah ada di OPEN maupun CLOSED, maka J dimasukkan ke OPEN. Sedangkan *node* D sudah ada di OPEN, maka harus di cek apakah *parent* dari D perlu diganti atau tidak. Ternyata, biaya dari S ke D melalui E (yaitu 10+12=22) lebih kecil daripada biaya dari S ke D, yaitu 38. Oleh karena itu, *parent* dari D harus di ubah, yang semula S menjadi E. Dengan perubahan *parent* ini, maka nilai g dan f pada D juga harus diperbarui (nilai g yang semula 38 menjadi 22, dan nilai f dari 135 menjadi 129). Langkah ke dua ini menghasilkan OPEN=[A,B,C,D,J] dan CLOSED=[S,E].



Gambar 3.4 Contoh *Graph* Langkah Ketiga

Fungsi evaluasi

f(G) = g(A) + g(A ke G) + h(G) = 20 +120 + 0 = 140

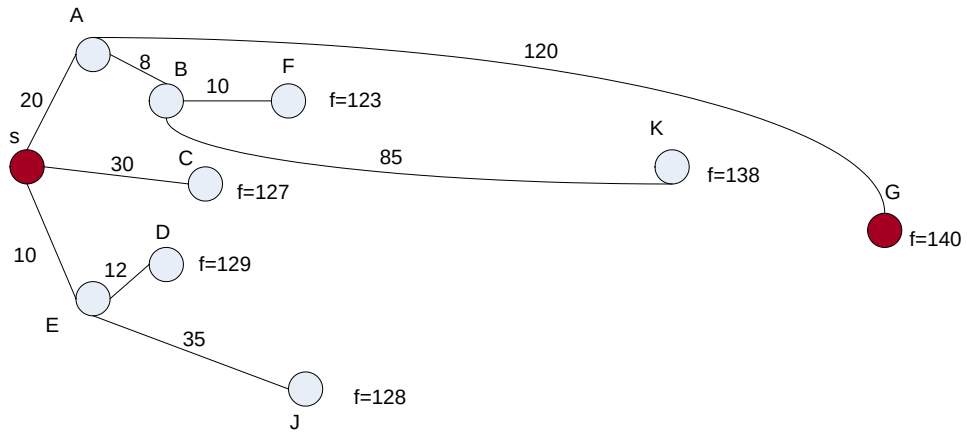
f(B) = g(A) + g(A ke B) + h(B) = 20 + 8 + 93 = 121

f(C) = 127

f(D) = 129

f(J) = 128

Lalu, A dengan biaya terkecil, yaitu 115, terpilih sebagai *bestnode* dan dipindahkan ke CLOSED. Selanjutnya, semua suksesor A dibangkitkan, yaitu : B dan G. Karena belum pernah ada di OPEN maupun di CLOSED, maka G dimasukkan ke OPEN. Sedangkan *node* B sudah ada di CLOSED, maka harus di cek apakah *parent* dari B perlu diganti atau tidak. Ternyata biaya dari S ke B yang melalui A, yaitu 20+8=28, lebih kecil daripada biaya dari S ke B (langsung), yaitu 35. Oleh karena itu, *parent* dari B harus diubah, agar didapat biaya terendah, yang semula S menjadi A. Nilai g dan f pada B juga harus diperbarui, nilai g yang semula 35 menjadi 28, nilai f dari 128 menjadi 121. Akhir dari langkah ini menghasikan OPEN=[B,C,D,J,G] dan CLOSED=[S,E,A].



Gambar 3.5 Contoh *Graph* Langkah Keempat

Fungsi Evaluasi :

f(G) = 140

f(F) = g(B) + g(B ke F) + h(F) = 28 + 10 + 85 = 123

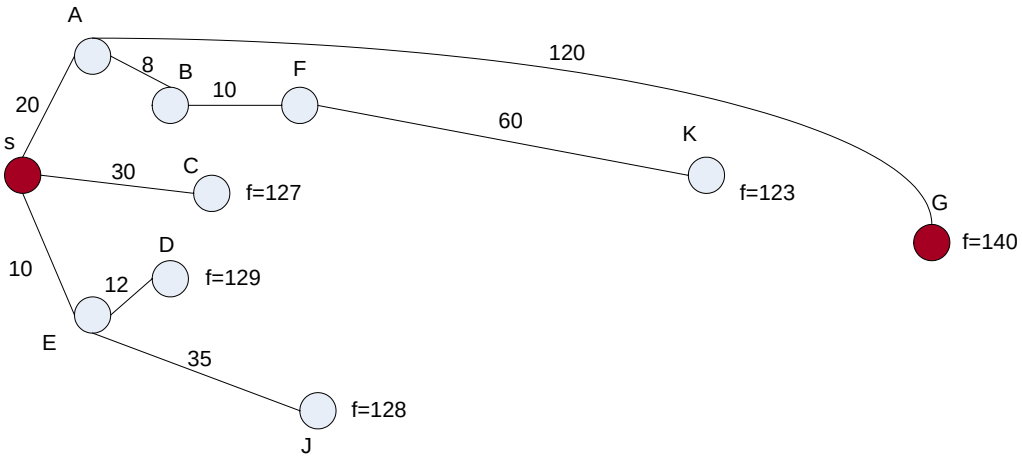
f(K) = g(B) + g(B ke K) + h(K) = 28 + 85 + 25 = 138

f(C) = 127

f(D) = 129

f(J) = 128

Karena *f(B)* memiliki biaya terkecil, yaitu 121, maka B terpilih sebagai *bestnode*, dan dipindahkan ke CLOSED. Lalu semua suksesor B dibangkitkan, yaitu F dan K. Karena F dan K belum pernah berada di OPEN maupun di CLOSED, maka F dan K dimasukkan ke OPEN. Akhir dari langkah ini menghasilkan OPEN=[C,D,F,J,G,K] dan CLOSED=[S,E,A,B].



Gambar 3.6 Contoh *Graph* Langkah Kelima

Fungsi Evaluasi :

f(G) = 140

f(K) = g(F) + g(F ke K) + h(K) = 38 + 60 +25 = 123

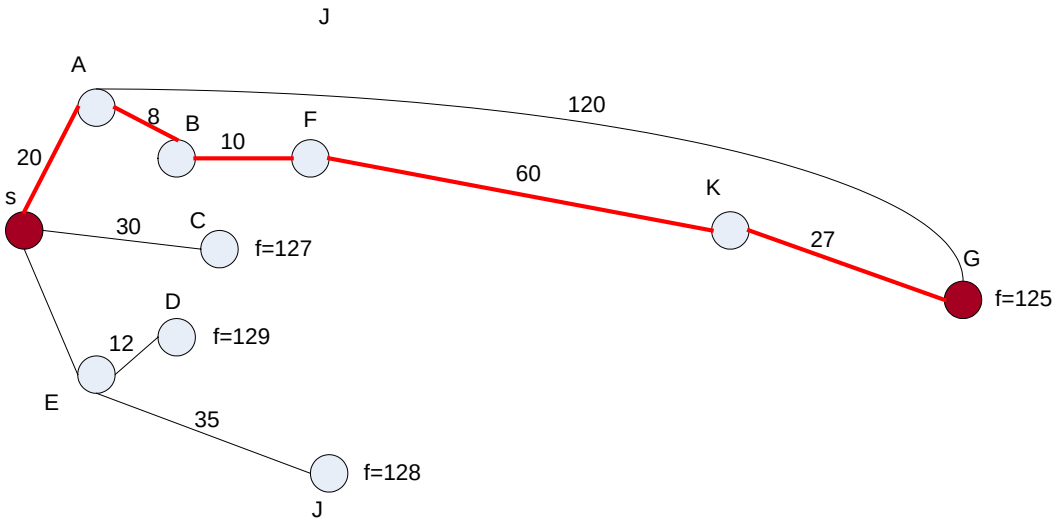
f(C) = 127

f(D) = 129

f(J) = 128

Langkah selanjutnya, F dengan biaya terkecil, yaitu 123, terpilih sebagai *bestnode* dan dipindahkan ke CLOSED. Selanjutnya, semua suksesor F dibangkitkan, yaitu K. karena K sudah ada di OPEN, maka harus di cek apakah *parent* dari K perlu diganti atau tidak. Biaya dari S ke K melalui F ternyata lebih kecil daripada biaya dari S ke K dengan B sebagai *parent*-nya. Oleh karena itu, *parent* dari K harus diubah, yang semula B menjadi F. Selanjutnya nilai g(K) yang semula 113 berubah menjadi 98, dan nilai f(K) yang semula 138 berubah menjadi 123. Akhirnya OPEN=[C, D, G, J, K] dan CLOSED=[S, E, A, B, F].

Berikutnya, K dengan biaya terkecil, yaitu 123 terpilih sebagai *bestnode* dan dipindahkan ke CLOSED. Selanjutnya, semua suksesor K dibangkitkan, yaitu *node* G. karena G sebelumnya telah berada di OPEN, maka terlebih dahulu harus diperiksa apakah *parent* dari G harus diganti atau tidak. Biaya dari S ke G melalui K ternyata lebih kecil daripada biaya dari S ke G melalui *parent* lama (A). Oleh karena itu, *parent* dari G harus diubah yang semula A menjadi K. Lalu nilai g(G) yang semula 140 diubah menjadi 125, dan nilai f(G) yang semula 140 menjadi 125. Dari langkah di atas didapat OPEN=[C,D,G,J] dan CLOSED=[S,E,A,B,F,K].



Gambar 3.7 Contoh *Graph* Langkah Keenam

Fungsi Evaluasi :

f(G) = g(K) + g(K ke G) + h(G) = 98 + 27+ 0 = 125

f(C) = 127

f(D) = 129

f(J) = 128

G dengan biaya terkecil, yaitu 125 terpilih sebagai *bestnode*. Karena *bestnode*-nya sama dengan goal, berarti solusi telah ditemukan. Lintasan dan total biaya biasa ditelusuri balik dari G menuju S, karena setiap *node* hanya memiliki satu *parent* dan setiap *node* memiliki informasi biaya yang sebenarnya (g). Penelusuran balik menghasilkan rute S-A-B-F-K-G dengan total jarak sama dengan 125, dan lintasan ini merupakan lintasan terpendek dari *node* S ke *node* G.

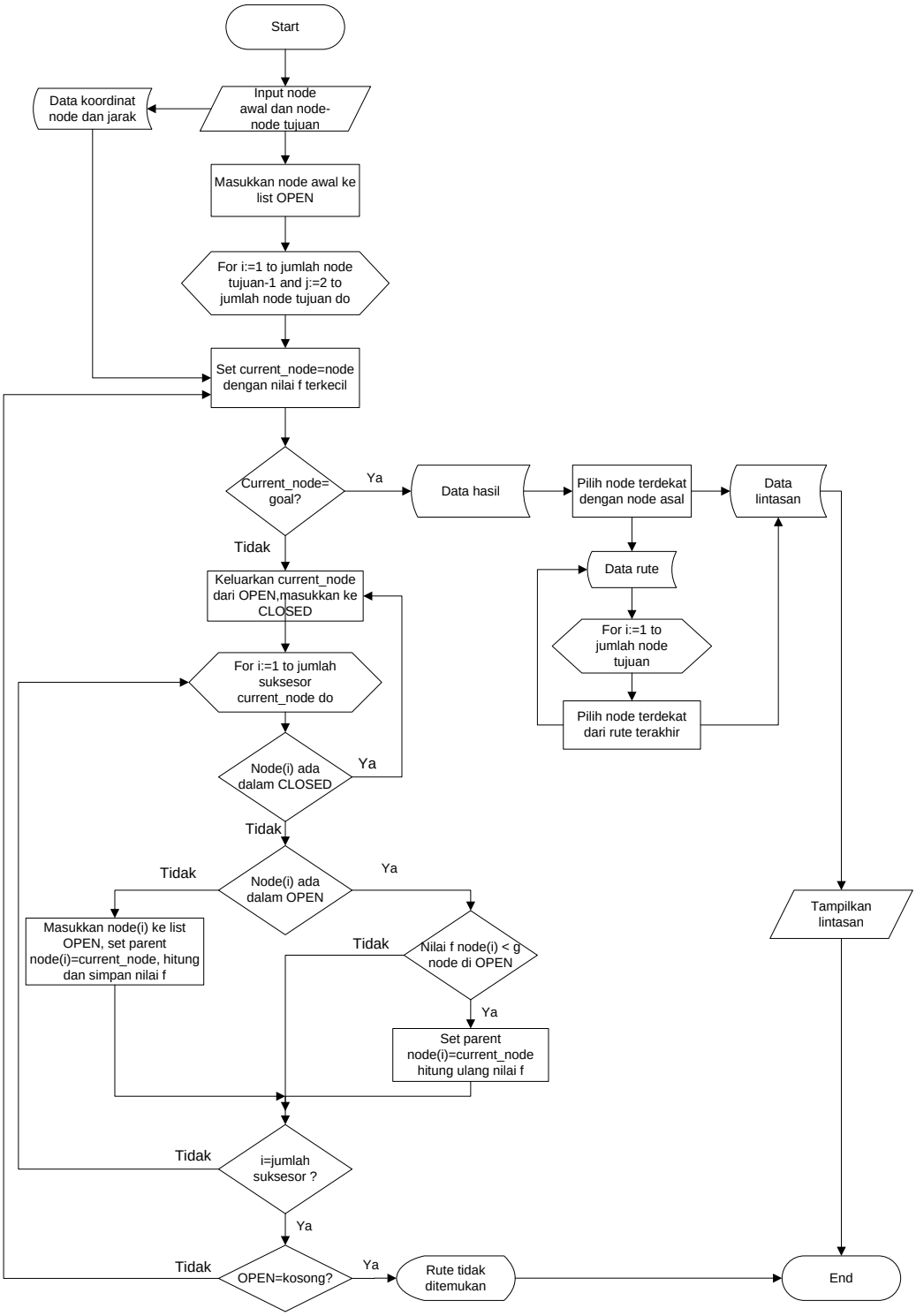
Pada kasus diatas A* membangkitkan dan menyimpan 10 (sepuluh) *node* dari 14 (empat belas) *node* yang ada pada *graph.*

3.2.1 *Flowchart* Algoritma A*

Flowchart adalah penggambaran secara grafik dari langkah-langkah dan urut-urutan prosedur dari suatu program. *Flowchart* menolong analis dan programmer untuk memecahkan masalah kedalam segmen-segmen yang lebih kecil dan menolong dalam menganalisis alternatif-alternatif lain dalam pengoperasian. *Flowchart* biasanya

mempermudah penyelesaian suatu masalah khususnya masalah yang perlu dipelajari dan dievaluasi lebih lanjut.

Secara umum langkah-langkah algoritma A* dalam sistem digambarkan pada flowchart dalam gambar 3.8 berikut:



Gambar 3.8 *Flowchart* algoritma A*

Dari *flowchart* pada Gambar 3.8 maka dapat dijelaskan sebagai berikut:

1. Buat *graph* pencari A, yang hanya berisi simpul awal, n_0 Taruh n_0 pada list bernama OPEN.
2. Buat sebuah list dengan nama CLOSED yang nilai awalnya kosong.
3. Jika OPEN tidak berisi, maka keluar dari program.
4. Ambil simpul pertama dari OPEN, hapus dari OPEN dan masukkan ke CLOSED. Anggap saja simpul n.
5. Jika n adalah simpul tujuan, keluar dari program dengan solusi yang diperoleh sepanjang pointer n ke n_0 pada A.
6. Perlebar simpul n, buat sebuah set M, dari suksesor yang bukan pendahulu di A.
7. Buat pointer kepada n dari tiap member M yang tidak ada pada A. Tambahkan anggota A pada OPEN. Untuk tiap anggota m, dari M yang sudah berada pada OPEN atau CLOSED, mengarahkan ulang pointer ke n jika jalur terbaik ke m yang ditemukan adalah melalui n. Untuk tiap anggota M yang telah berada pada CLOSED, arahkan ulang pointer dari tiap pendahulu di A sehingga mereka mengarahkan mundur sepanjang jalan terbaik yang telah ditemukan.
8. Susun ulang daftar OPEN dengan urutan nilai f yang menaik.
9. Kembali ke langkah nomor 3.

3.3 Deskripsi Sistem

Tugas Akhir ini akan membangun sebuah *prototype* sistem yang menunjukkan lintasan terpendek antara beberapa *node*, yang diimplementasikan kepada suatu sistem informasi geografis. Sistem ini dapat digunakan oleh ekspedisi yang membutuhkan petunjuk terhadap lintasan terpendek pengantaran barang di Kota Medan.

Pada sistem ini akan diberikan beberapa *node-node* berupa persimpangan jalan yang ada di peta Kota Medan. *User* kemudian memilih titik-titik node yang akan dikunjungi, kemudian sistem akan melakukan perhitungan dan menentukan jalur terpendek yang harus dilalui mulai dari kantor sebagai titik awal hingga node terakhir yang dikunjungi.

3.4 Spesifikasi Keperluan Sistem

Dalam tugas akhir ini, sistem akan dibangun sebagai pencari lintasan terpendek yang mengimplementasikan algoritma A*. Spesifikasi umum kebutuhan sistem menjelaskan dasar pembuatan rancangan sistem yang terdiri dari fungsi sistem, tujuan sistem, masukan dan keluaran sistem, dan batasan sistem.

3.4.1 Fungsi Sistem

Sistem yang akan dibuat memiliki fungsionalitas berikut:

1. Menampilkan peta Kota Medan.
2. Menunjukkan *node-node* yang merupakan beberapa persimpangan di Kota Medan.
3. Menghitung jarak terpendek antara semua *node*.
4. Menampilkan jalur terpendek pada peta.

3.4.2 Tujuan Sistem

Sistem yang akan dibuat memiliki tujuan untuk menghitung dan menunjukkan lintasan terpendek dari *node* awal ke beberapa *node* tujuan pada peta.

3.4.3 Masukan dan Keluaran Sistem

Masukan (*input*) sistem berupa data titik-titik yang akan dipilih dari *node-node* yang telah ditentukan saat inisialisasi peta sedangkan keluaran (*output*) sistem adalah jalur-jalur yang dilalui *shortest path* dari data awal ke data akhir pada peta.

3.4.4 Batasan Sistem

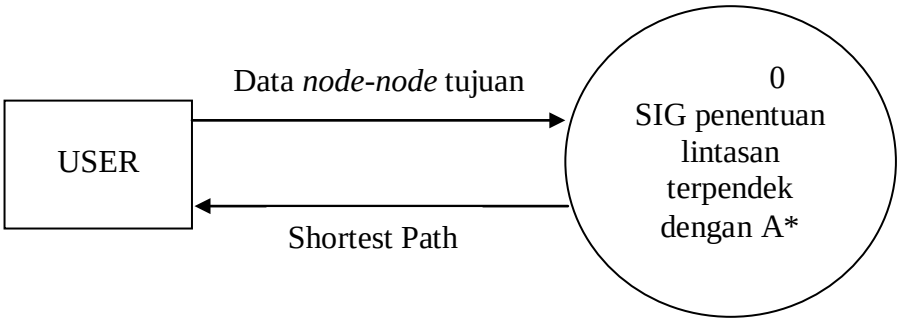
Sistem yang akan dibuat memiliki batasan-batasan berikut:

1. Titik awal lintasan sudah ditentukan.
2. *Node* yang dapat dipilih sebagai titik yang ingin dikunjungi sudah tertentu.
3. Peta yang ditampilkan hanya peta Kota Medan.

3.5 *Data Flow Diagram* (DFD)

DFD (*Data Flow Diagram*) merupakan suatu model logika data atau proses yang dibuat untuk menggambarkan dari mana asal data dan kemana tujuan data yang keluar dari sistem, dimana data disimpan, proses apa yang menghasilkan data tersebut dan interaksi antara data yang tersimpan dan proses yang dikenakan pada data tersebut. Model fungsional ini berfungsi membantu memahami cara kerja sistem dan hubungan setiap proses dalam sistem secara terstruktur dan logis.

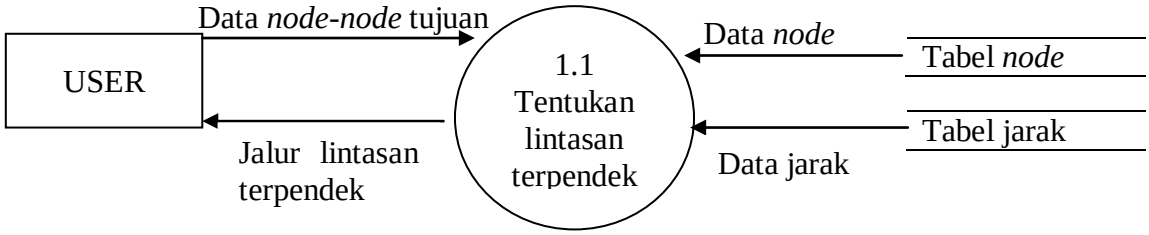
Diagram konteks adalah diagram yang terdiri dari suatu proses dan menggambarkan ruang lingkup suatu sistem. Diagram konteks merupakan DFD level 0 yang menggambarkan seluruh input ke sistem atau output dari sistem.



Gambar 3.9 Diagram konteks

Diagram konteks berfungsi menggambarkan hubungan antara entitas luar, berupa masukan dan keluaran sistem. Dari gambar 3.9 dapat dilihat bahwa hanya terdapat sebuah entitas luar, yaitu *user*. *User* disini bertindak sebagai *source terminal* yang memberikan masukan ke sistem dan juga sebagai *sink terminal* yang menerima keluaran dari sistem itu sendiri.

Berdasarkan diagram konteks pada gambar 3.9, DFD level 1 hanya memiliki satu proses yaitu proses untuk menentukan lintasan terpendek, maka DFD level 1 dapat digambarkan sebagai berikut:



Gambar 3.10 DFD Level 1

Pada proses Tentukan lintasan terpendek membutuhkan data masukan berupa data *node-node* tujuan, kemudian proses akan mengambil data dari tabel *node* dan tabel jarak. Lalu sistem akan melakukan pencarian lintasan terpendek dengan menggunakan algoritma A*. Proses-proses yang terjadi pada DFD Level 1 diuraikan pada tabel 3.2 di bawah ini.

Tabel 3.2 Spesifikasi Proses DFD Level 1

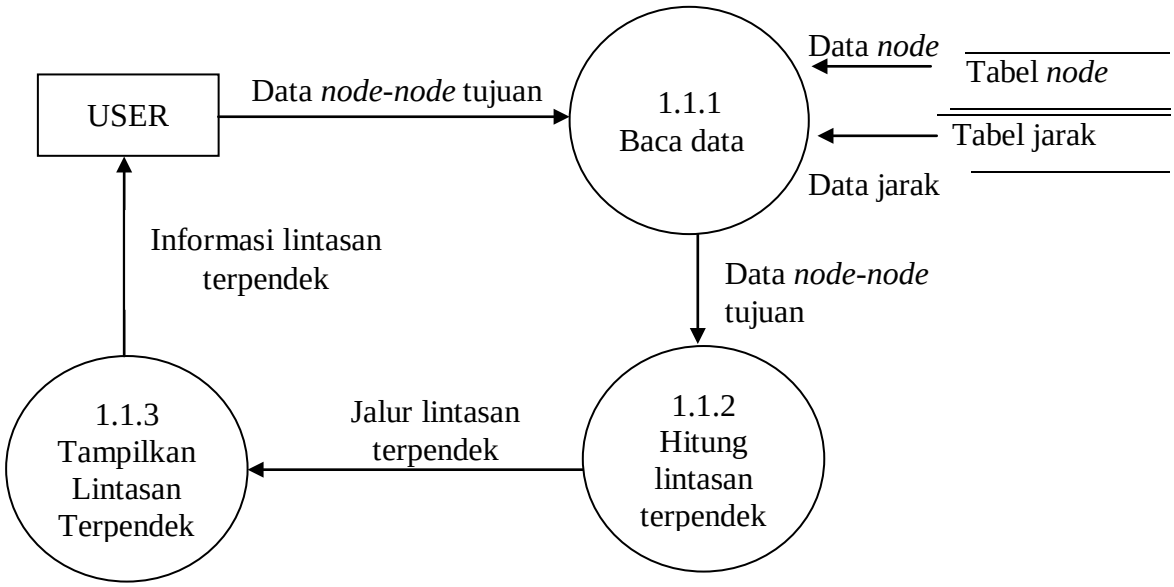
No./Nama Proses	Input	Keterangan Proses	Output
1.1/Tentukan lintasan terpendek	Data *node-node* tujuan	Menentukan lintasan terpendek berdasarkan inputan berupa *node-node* tujuan yang tersimpan dalam *database*	Jalur lintasan terpendek

Masing-masing entitas data yang tercantum pada DFD level 1 ditampilkan pada tabel di bawah ini.

Tabel 3.3 Entitas Data Pada DFD Level 1

Nama	Keterangan
Data *node-node* tujuan	*Node* yang dipilih oleh user untuk dikunjungi
Jalur lintasan terpendek	Hasil perhitungan lintasan terpendek
Data *node*	Data koordinat *node*
Data jarak	Data jarak antar *node*

Proses tentukan lintasan terpendek terdiri dari dua proses, yaitu proses baca data dan proses hitung lintasan terpendek. DFD level 2 untuk proses tentukan lintasan terpendek dapat digambarkan sebagai berikut:



Gambar 3.11 DFD Level 2

Proses baca data (1.1.1) berfungsi membaca data dari *database*. Proses ini membaca data *node* dan *edge* dari *database* dan membaca data jalur *shortest path* hasil proses hitung lintasan terpendek (1.1.2) akan ditampilkan pada user pada proses tampilkan lintasan terpendek (1.1.3). Proses-proses yang terjadi pada DFD Level 2 diuraikan pada tabel 3.4 di bawah ini.

Tabel 3.4 Spesifikasi Proses DFD Level 2

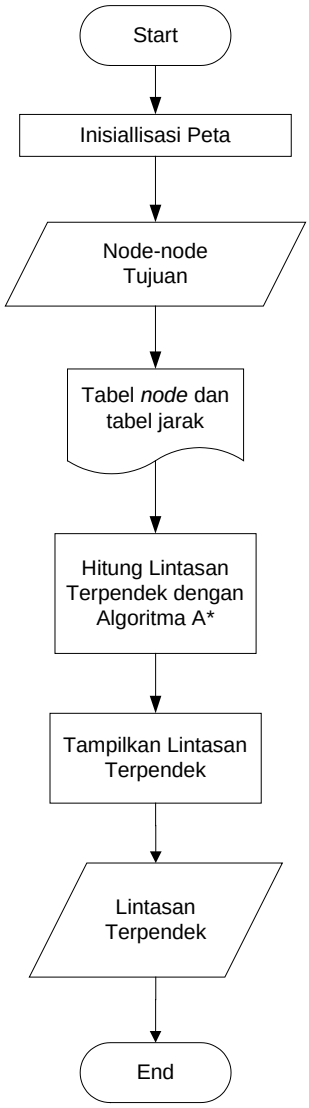
No./Nama Proses	Input	Keterangan Proses	Output
1.1.1/Proses Baca Data	Data *node-node* tujuan, data *node*, data jarak	Proses membaca data masukkan yang diberikan oleh pengguna yang tersimpan dalam database	Data *node-node* tujuan
1.1.2/Proses Hitung Lintasan Terpendek	Data *node-node* tujuan	Proses perhitungan lintasan terpendek menggunaan Algoritma A*	Jalur lintasan terpendek
1.1.3/Proses Tampilkan Lintasan Terpendek	Jalur lintasan terpendek	Proses untuk menampilkan pada peta hasil perhitungan lintasan terpendek	Informasi lintasan terpendek

Masing-masing entitas data yang tercantum pada DFD level 2 ditampilkan pada tabel di bawah ini.

Tabel 3.5 Entitas Data Pada DFD Level 2

Nama	Keterangan
Data *node-node* tujuan	*Node* yang dipilih oleh *user* untuk dikunjungi
Data *node*	Data yang tersimpan dalam tabel *node*
Data jarak	Data yang tersimpan dalam tabel jarak
Jalur lintasan Terpendek	Hasil dari perhitungan pencarian lintasan terpendek dengan menggunakan Algoritma A*
Informasi lintasan terpendek	Jarak dan lintasan terpendek dalam peta berupa garis yang menunjukkan lintasan terpendek

Secara umum alur proses sistem dapat dilihat pada *flowchart* sebagai berikut:



Gambar 3.12 *Flowchart* Sistem

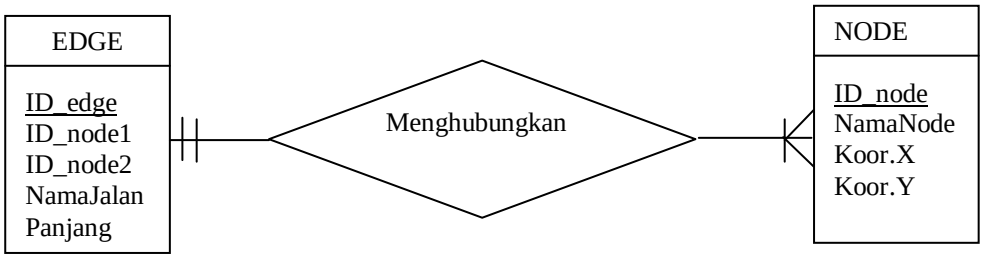
Secara garis besar, Aplikasi Sistem Informasi Geografis Penentuan Lintasan Terpendek Pengantaran Barang Menggunakan Algoritma A* dan Metode Hamilton ini dapat dituliskan sebagai berikut:

1. Mulai.
2. Inisialisasi peta.
3. Membaca data seluruh *node* dan bobot sisi pada *database node* dan jarak.
4. Input *node-node* tujuan.
5. Melakukan proses perhitungan lintasan terpendek menggunakan Algoritma A* berdasarkan *input* dari langkah sebelumnya.
6. Menampilkan lintasan terpendek pada peta.
7. Selesai.

3.6 Desain *Database*

Pada sistem ini menggunakan aplikasi *database Microsoft Office Access* 2007, yaitu aplikasi basis data komputer relasional yang berjalan dibawah sistem windows yang dapat menangani database dengan skala besar maupun kecil..

Entity Relationship Diagram (ERD) adalah pemodelan data utama dan akan membantu mengorganisasikan data dalam suatu proyek ke dalam entitas-entitas dan menentukan hubungan antar entitas. Model struktur data dan hubungan antar data pada sistem ini dapat dilihat pada gambar berikut :



Gambar 3.13 ERD Sistem

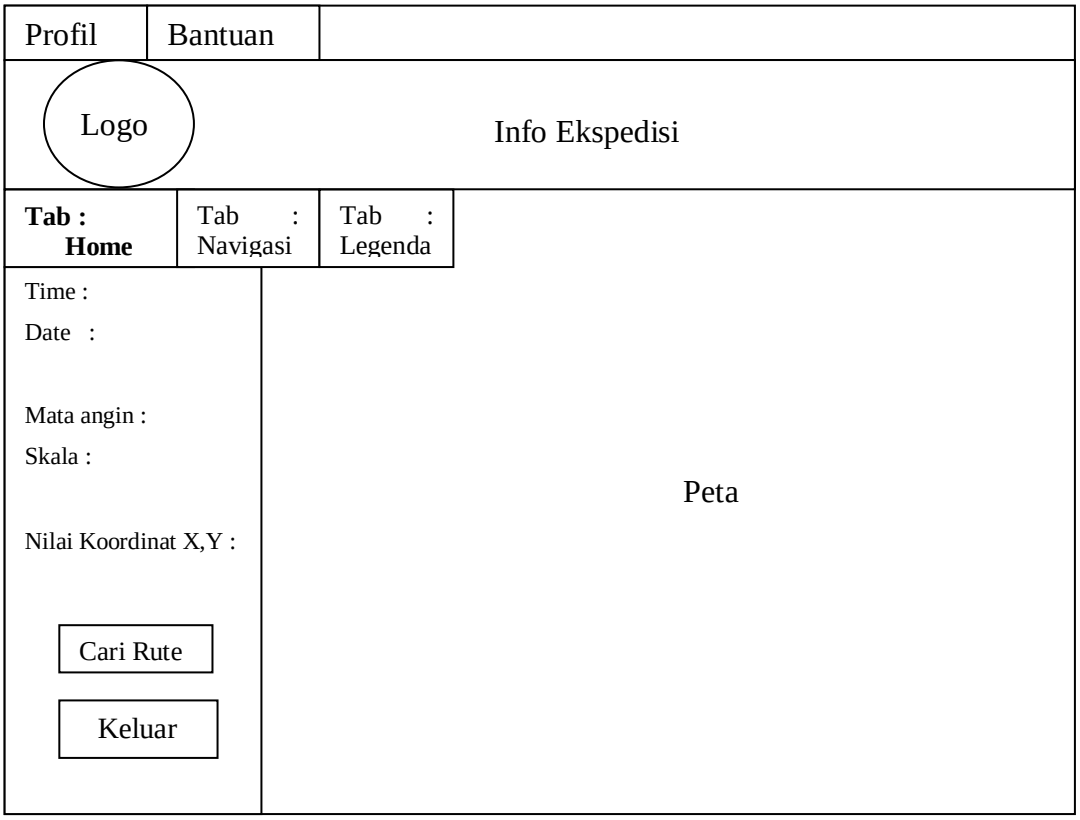
3.7 Perancangan Antarmuka Sistem (*Interface*)

Rancangan antarmuka dari sistem ini terdiri dari 4 form yaitu, form utama, form shortestpath, form profil dan form bantuan. Setiap form dirancang untuk memudahkan perancang untuk mengimplementasikan antarmuka dan memudahkan user untuk memahami cara penggunaan aplikasi. Berikut ini rancangan setiap form yang akan digunakan pada aplikasi Sistem Informasi Geografis Penentuan Lintasan Terpendek Pengantaran Barang Menggunakan Algoritma A*.

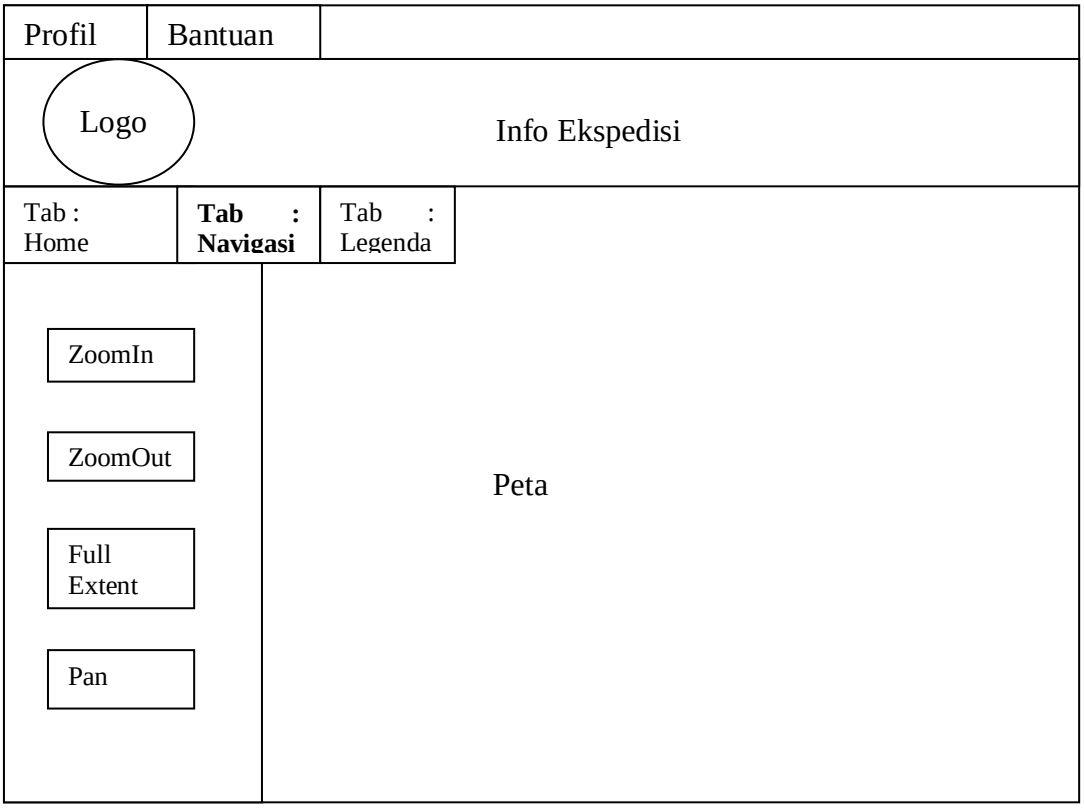
1. Form Utama

   Form ini merupakan form yang pertama kali muncul saat user menjalankan aplikasi. Form ini memuat dua menu yaitu, profil dan bantuan. Pada form ini, user dapat melihat peta Kota Medan secara menyeluruh dan pada form ini tersedia *button-button navigasi* peta, informasi titik-titik pada peta, legenda, mata angin

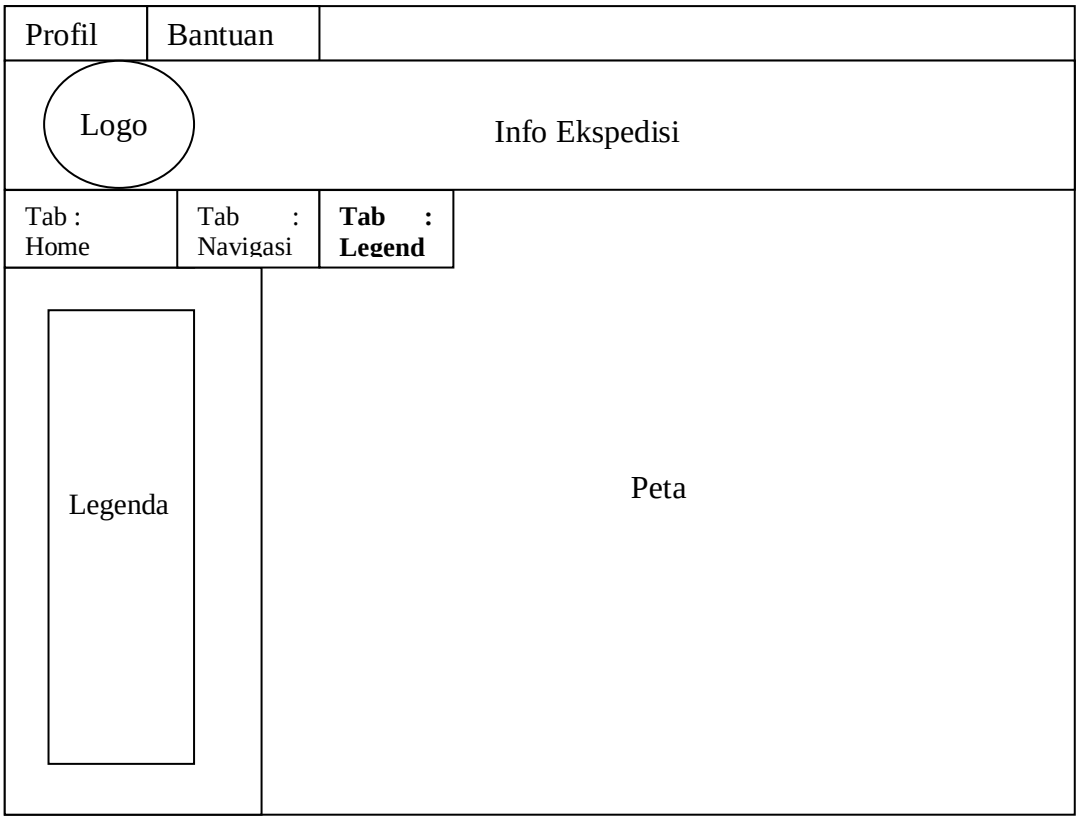
dan skala. Untuk keluar dari aplikasi, user dapat mengklik *button* keluar yang tersedia pada padaa form ini. Berikut adalah tampilan form utama :



Gambar 3.14 Rancangan Form Utama Tab *Home*



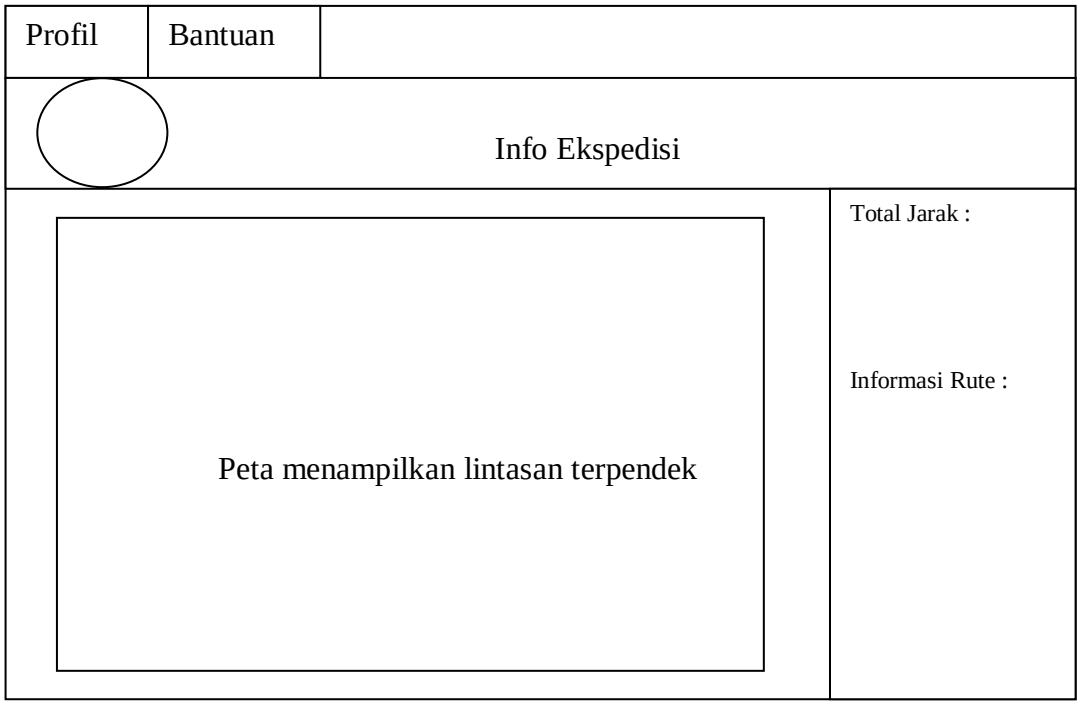
Gambar 3.15 Rancangan Form Utama Tab *Navigasi*



Gambar 3.16 Rancangan Form Utama Tab Legenda

2. Form *Shortestpath*

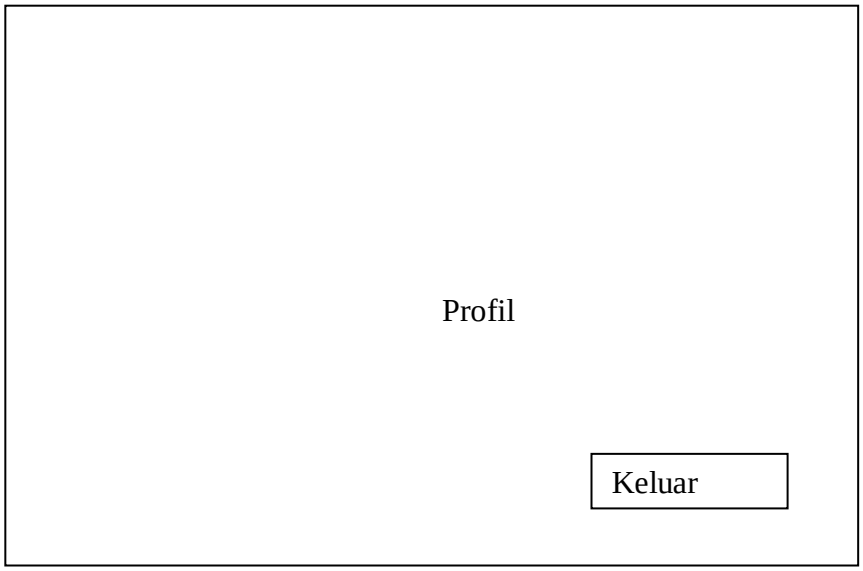
   Form ini akan menampilkan lintasan terpendek yang dicari dengan algoritma A* dan informasi mengenai jarak dan titik-titik yang ditempuh dalam lintasan.



Gambar 3.17 Rancangan Form *Shortestpath*

3. Form Profil

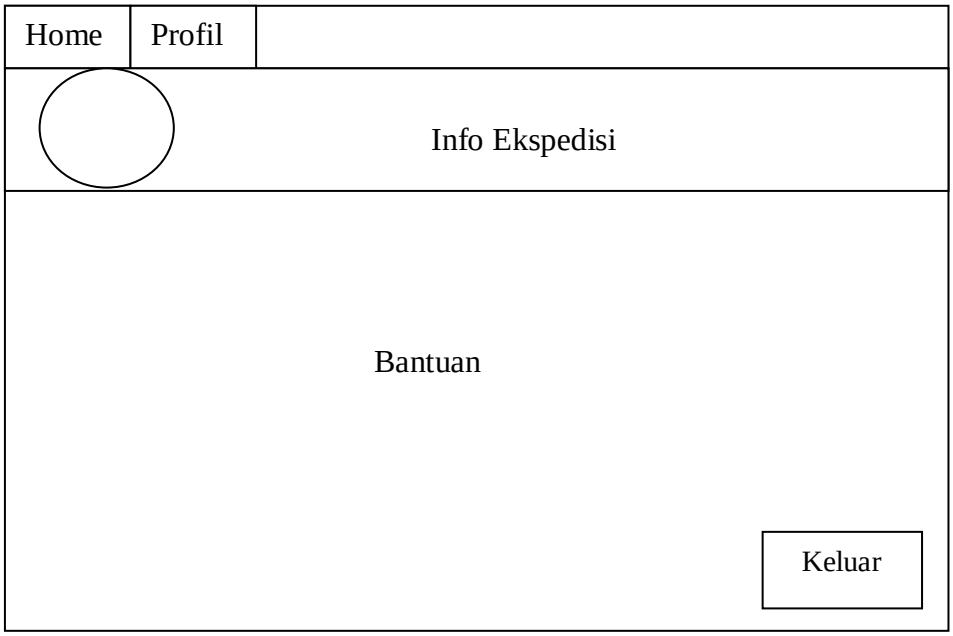
   Form ini menampilkan informari mengenai profil pembuat sistem



Gambar 3.18 Rancangan Form Profil

4. Form Bantuan

Form ini berisikan tentang tata cara penggunaan aplikasi. Hal ini bertujuan untuk memandu user jika mengalami kesulitan dalam menggunakan aplikasi.



Gambar 3.19 Rancangan Form Bantuan

BAB 4

IMPLEMENTASI DAN PENGUJIAN SISTEM

4.1 Implementasi

Pada tahap implementasi, seluruh perancangan telah diwujudnyatakan dalam suatu bahasa pemrograman. Seluruh kriteria dan pemodelan yang telah didefenisikan sebelumnya akan menjadi acuan dalam proses implementasi ini.

Aplikasi ini bertujuan untuk menunjukkan lintasan terpendek pada sistem informasi geografis yang memuat peta Kota Medan. Aplikasi ini hanya menunjukkan jalur terpendek dari jalan protokol pada Kota Medan dan diimplementasikan dengan berbasis *desktop*.

Pada aplikasi ini terdapat empat *form*, yaitu *form* utama,*form shortestpath, form* profil dan *form* bantuan. *Form-form* tersebut akan dijelaskan pada bab ini.

4.1.1 Lingkungan Implementasi

Lingkungan implementasi yang akan dijelaskan merupakan lingkungan perangkat keras (*hardware*) dan perangkat lunak (*software*) yang digunakan dalam penulisan skripsi ini. Spesifikasi perangkat keras yang digunakan adalah sebagai berikut :

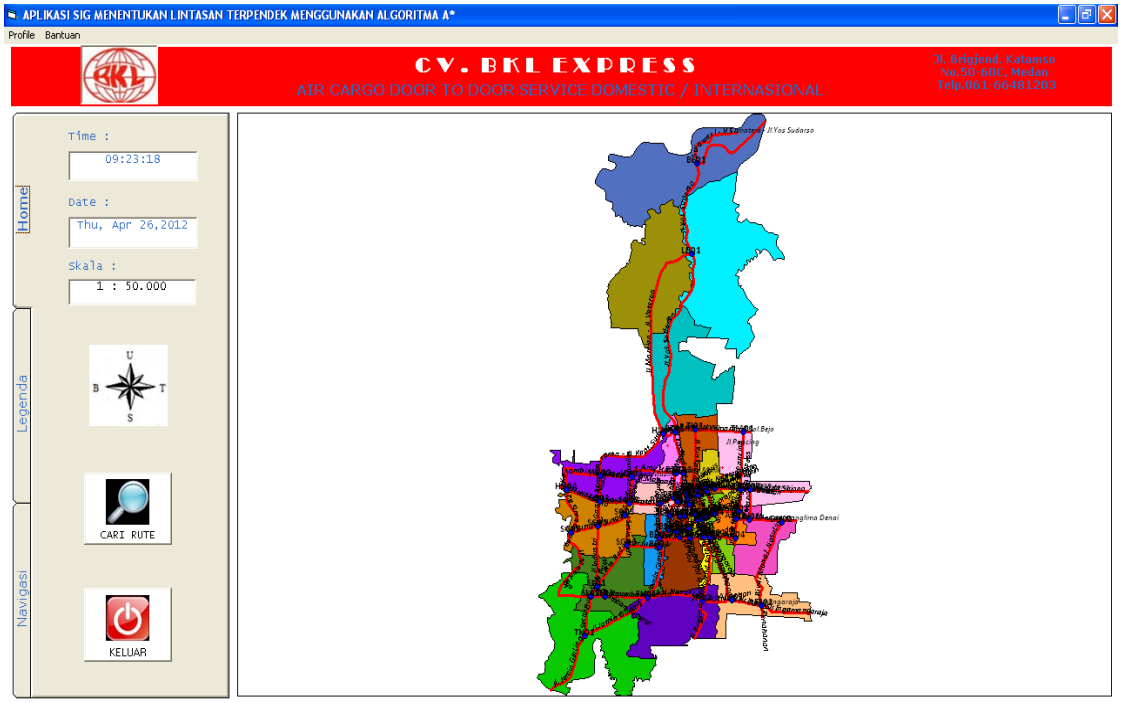
1. *Processor* Intel® Core™ i3 2.13GHz
2. *Memory* RAM 2 GB
3. *Harddisk* 500 GB
4. Perangkat *output* berupa monitor CRT 14”
5. Perangkat *input* berupa *mouse* dan *keyboard*

Spesifikasi perangkat lunak yang digunakan adalah sebagai berikut :

1. *Operating system Microsoft Windows XP*
2. *Microsoft Visual Basic 6.0*
3. *ArcView GIS 3.0*
4. *MapObject 2.1*
5. *Microsoft Office Access 2007*

4.1.2 Tampilan *Home*

Form ini merupakan halaman pertama yang akan muncul ketika *user* menjalankan aplikasi. Pada *form* ini *user* dapat melihat tampilan peta Kota Medan, jalan protokol, informasi nama jalan, dan informasi *node.* Pada *form* ini juga tersedia dua buah menu yaitu profil dan bantuan, tiga buah *tab* yaitu *Home*, *Navigasi* dan Legenda. Tampilan *home* dapat dilihat pada gambar berikut :



Gambar 4.1 Tampilan *Form* Utama

Untuk mulai melakukan pencarian lintasan terpendek, *user* dapat memilih *butto* CARI RUTE pada tab *Home.* Jika *user* ingin memperjelas tampilan peta, disediakan *button* navigasi pada *tab* Navigasi. *Button* navigasi terdiri dari *Zoom In,*

Zoom Out, Full Extent dan *Pan*. Selain itu, pada tab navigasi juga ditampilkan informasi mengenai koordinat UTM dari posisi pointer mouse pada permukaan bumi sesungguhnya.

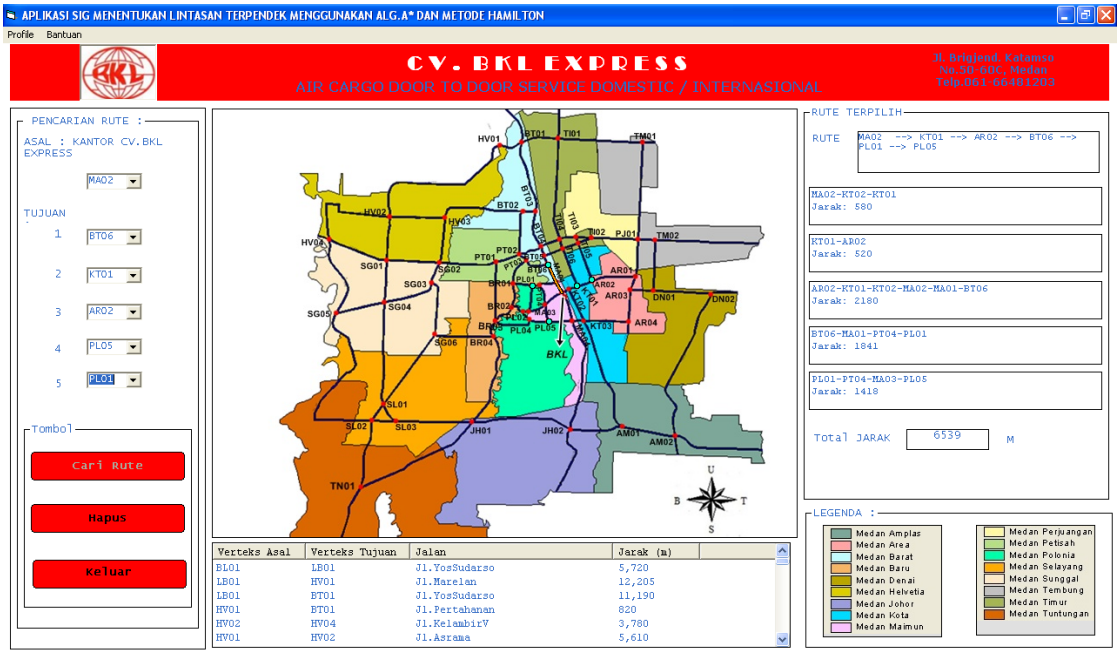
Untuk melakukan pencarian lintasan terpendek kembali, *user* dapat mengklik *button* TUTUP untuk kembali ke form Utama. Untuk mengakhiri aplikasi, *user* dapat mengklik *button* KELUAR.

4.1.3 Tampilan *Shortestpath*

Form ini user dapat melakukan penencari rute terpendek. Untuk mulai melakukan pencarian, *user* terlebih dahulu menginputkan *node-node* tujuan yang tersedia pada *combobox* tujuan. Sedangkan untuk *node* awal sudah ditentukan yaitu kantor CV. BKL EXPRESS yang berada di Jl.Brigjend. Katamso, Medan Maimun. Hal ini disebabkan karena semua titik awal keberangkatan kurir adalah kantor tersebut. Untuk dapat memahami kode-kode persimpangan jalan, user dapat melihat *node* pada peta. Setelah memasukkan node-node tujuan, user dapat mengklik *button* CARI RUTE untuk melakukan pencarian lintasan terpendek menggunakan algoritma A*. Hasil lintasan terpendek akan ditampilkan pada peta form *Shortestpath*. Selain itu total jarak dan dan urutan *node-node* yang dikunjungi akan ditampilkan pada bagian kanan. Hal ini memudahkan *user* untuk melihat informasi lintasan terpendek.

Dalam satu kali proses pencarian lintasan terpendek, *user* tidak dapat memilih *node* tujuan yang sama berulang kali. Jika *user* menginput *node* tujuan yang sama berulang kali, maka akan combobox tujuan berikutnya tidak akan aktif sampai *user* mengganti *node* tujuan yang berbeda.

Untuk melakukan pencarian rute terpendek kembali, *user* terlebuh dahulu memilih *button* HAPUS untuk membersihkan peta dan *combobox* dari pencarian rute sebelumnya.



Gambar 4.2 Tampilan *Form Shortestpath*

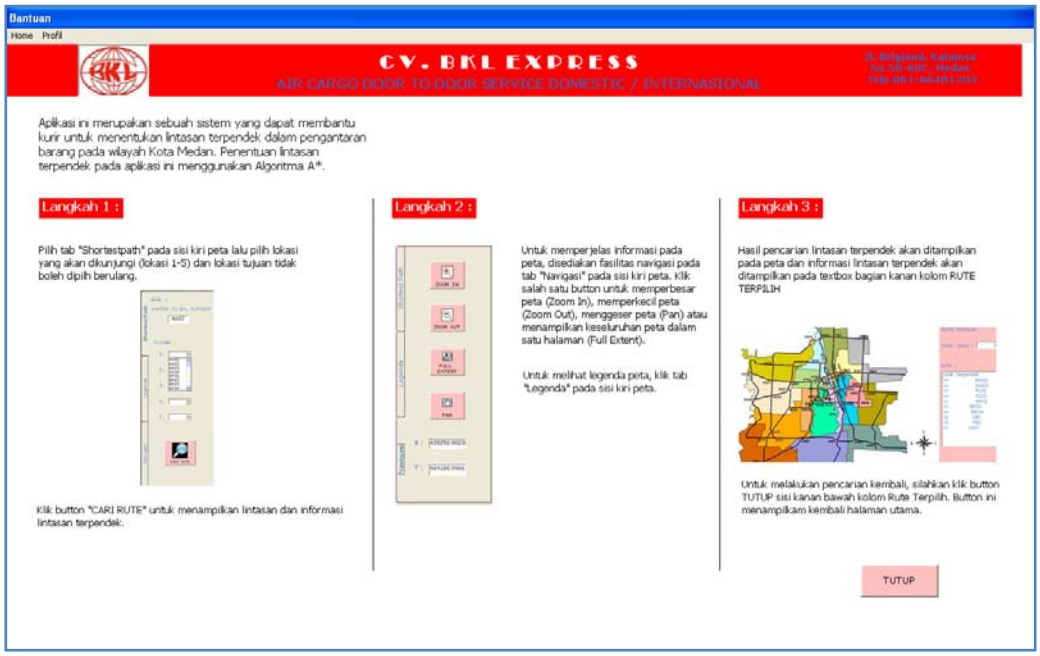
4.1.4 Tampilan Profil

Form ini akan menampilkan informasi mengenai profil pembuat sistem. Tampilan profil dapat dilihat pada gambar berikut :

Gambar 4.3 Tampilan *Form* Profil

4.1.5 Tampilan Bantuan

Pada *form* ini akan dijelaskan tentang cara kerja dan tata cara penggunaan aplikasi. Diharapkan *form* ini akan membantu user untuk memahami dan menggunakan aplikasi. Berikut adalah gambar tampilan bantuan :



Gambar 4.4 Tampilan *Form* Bantuan

4.2 Pengujian Sistem

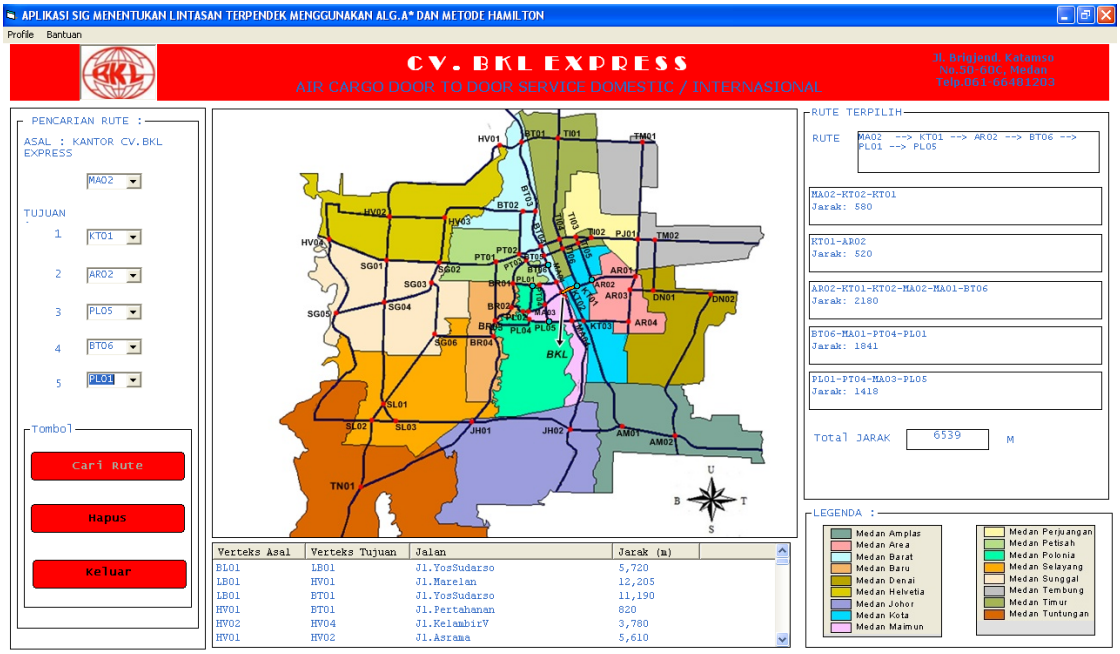
Dalam tahap pengujian akan dilakukan validasi terhadap perangkat lunak yang telah dihasilkan. Validasi perangkat lunak ditujukan untuk menunjukkan bahwa aplikasi sesuai dengan spesifikasinya dan bahwa sistem memenuhi harapan penggunan aplikasi.

Pengujian sistem ini dilakukan untuk melihat apakah algoritma A* dapat menentukan lintasan terpendek dengan optimal. Pengujian ini dilakukan dengan mengubah urutan inputan *node-node* tujuan. Pengujian *shortest path* dengan urutan inputan :

1. KTO1 – AR02 – PL05 – BT06 – PL01
2. AR02 – PL05 – KT01 – BT06 – PL01
3. PL05 – KT01 – AR02 – PL01 – BT06
4. PL01 – PL05 – BT06 – AR02 – PL01

4.2.1 Pengujian *Shortest Path* dengan Urutan Inputan I

Pada pengujian ini dipilih urutan inputan node :KTO1 – AR02 – PL05 – BT06 – PL01



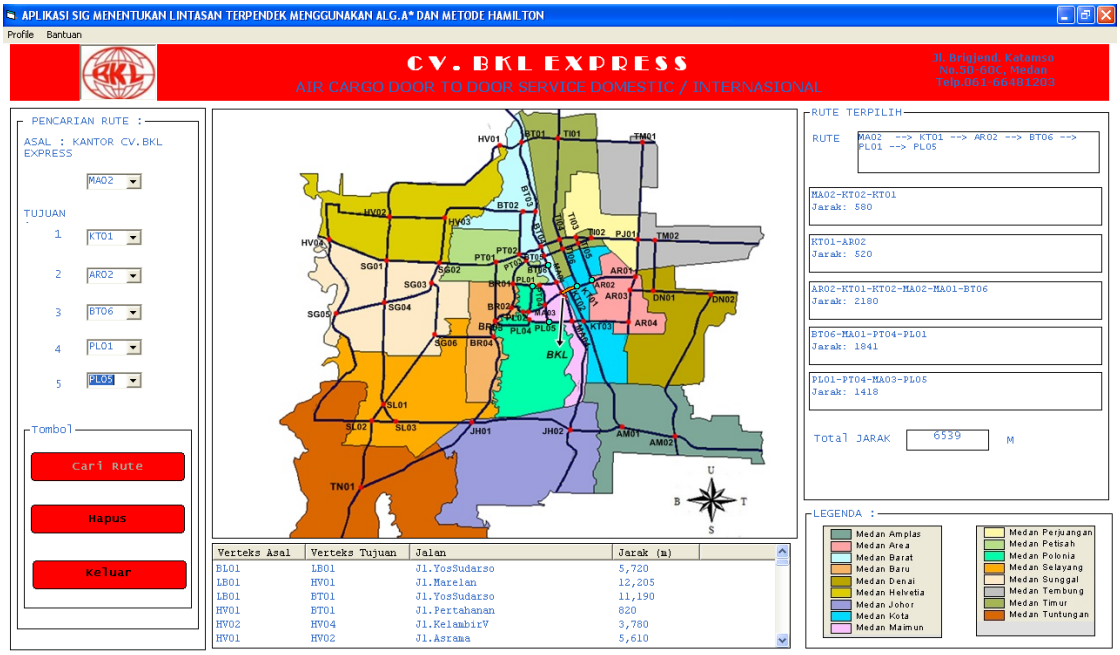
Gambar 4.5 Tampilan Lintasan Pengujian 1

Hasil perhitungan lintasan terpendek dengan algoritma A* menghasilkan urutan kunjungan : MA02 – KT01 – AR02 – BT06 – PL01 – PL05 dan total jarak yang ditempuh adalah 6.349 m. Dengan rincian node yang dikunjungi yaitu :

1. MA02 – KT02 – KT01 dengan jarak 580 m
2. KT01 – AR01 dengan jarak 520 m
3. AR02 – KT01 – KT02 – MA02 – MA01 – BT06 dengan jarak 2.180 m
4. BT06 – MA01 – PT04 – PL01 dengan jarak 1.841 m
5. PL01 – PT04 – MA03 – PL05 dengan jarak 1.418 m

4.2.2 Pengujian *Shortest Path* dengan Urutan Inputan II

Pada pengujian ini dipilih urutan inputan node : AR02 – PL05 – KT01 – BT06 – PL01



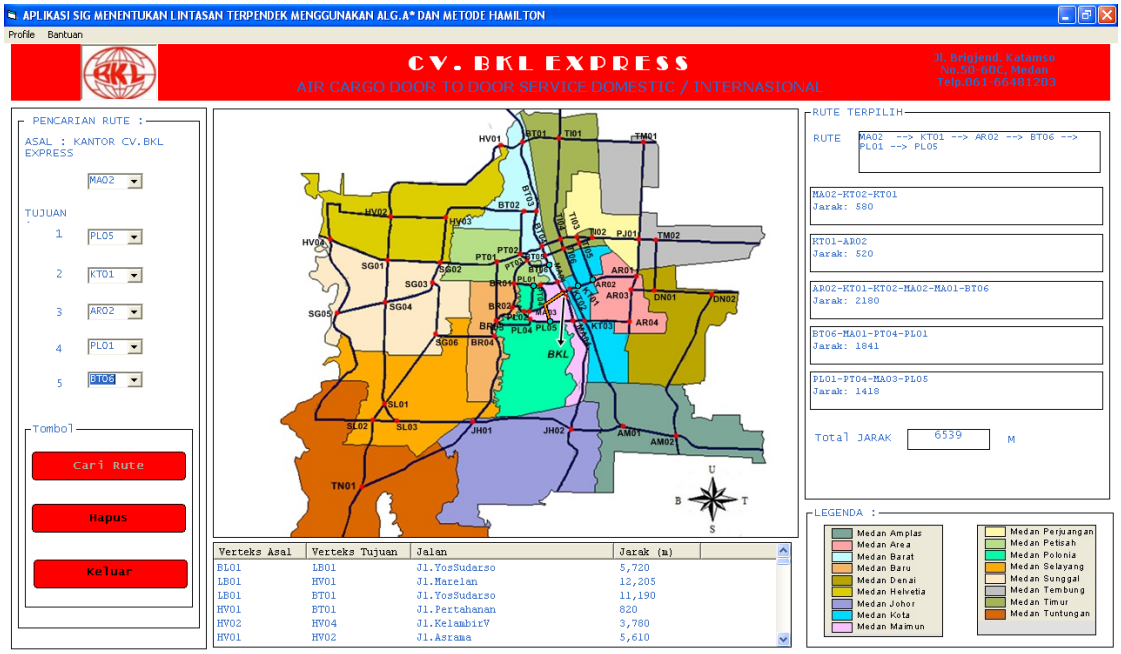
Gambar 4.6 Tampilan Lintasan Pengujian II

Hasil perhitungan lintasan terpendek dengan algoritma A* menghasilkan urutan kunjungan : MA02 – KT01 – AR02 – BT06 – PL01 – PL05 dan total jarak yang ditempuh adalah 6.349 m. Dengan rincian node yang dikunjungi yaitu :

1. MA02 – KT02 – KT01 dengan jarak 580 m
2. KT01 – AR01 dengan jarak 520 m
3. AR02 – KT01 – KT02 – MA02 – MA01 – BT06 dengan jarak 2.180 m
4. BT06 – MA01 – PT04 – PL01 dengan jarak 1.841 m
5. PL01 – PT04 – MA03 – PL05 dengan jarak 1.418 m

4.2.3 Pengujian *Shortest Path* dengan Urutan Inputan III

Pada pengujian ini dipilih urutan inputan node : PL05 – KT01 – AR02 – PL01 – BT06



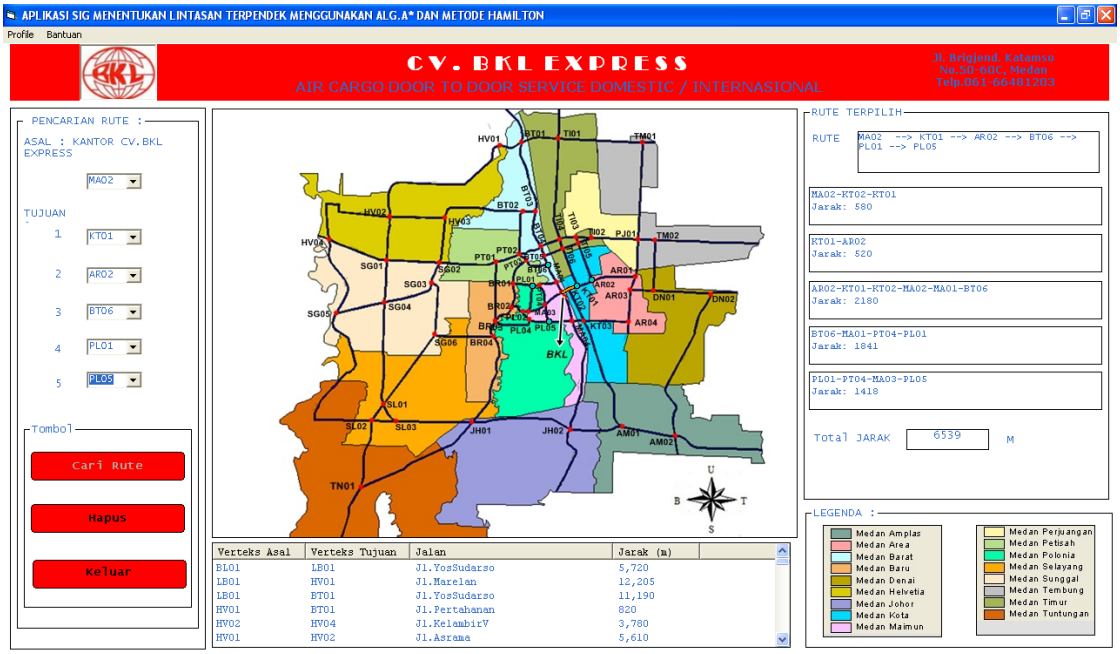
Gambar 4.7 Tampilan Lintasan Pengujian III

Hasil perhitungan lintasan terpendek dengan algoritma A* menghasilkan urutan kunjungan : MA02 – KT01 – AR02 – BT06 – PL01 – PL05 dan total jarak yang ditempuh adalah 6.349 m. Dengan rincian node yang dikunjungi yaitu :

1. MA02 – KT02 – KT01 dengan jarak 580 m
2. KT01 – AR01 dengan jarak 520 m
3. AR02 – KT01 – KT02 – MA02 – MA01 – BT06 dengan jarak 2.180 m
4. BT06 – MA01 – PT04 – PL01 dengan jarak 1.841 m
5. PL01 – PT04 – MA03 – PL05 dengan jarak 1.418 m

4.2.4 Pengujian *Shortest Path* dengan Urutan Inputan IV

Pada pengujian ini dipilih urutan inputan node : PL01 – PL05 – BT06 – AR02 – PL01



Gambar 4.8 Tampilan Lintasan Pengujian IV

Hasil perhitungan lintasan terpendek dengan algoritma A* menghasilkan urutan kunjungan : MA02 – KT01 – AR02 – BT06 – PL01 – PL05 dan total jarak yang ditempuh adalah 6.349 m. Dengan rincian node yang dikunjungi yaitu :

1. MA02 – KT02 – KT01 dengan jarak 580 m
2. KT01 – AR01 dengan jarak 520 m
3. AR02 – KT01 – KT02 – MA02 – MA01 – BT06 dengan jarak 2.180 m
4. BT06 – MA01 – PT04 – PL01 dengan jarak 1.841 m
5. PL01 – PT04 – MA03 – PL05 dengan jarak 1.418 m

Dari empat kali pengujian sistem dengan memberikan nilai inputan *node-node* yang sama dan urutan yang berbeda, sistem tetap memberikan *output* yang sama. Sehingga urutan inputan tidak mempengaruhi *output* karena sistem akan memilih dan menampilkan lintasan terpendek yang optimal.

BAB 5

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan pembahasan dan evaluasi dari bab-bab terdahulu dan teori yang ada, maka dapat ditarik kesimpulan sebagai berikut:

1. Algoritma A* terbukti dapat menyelesaikan masalah kurir dalam menentukan lintasan terpendek dalam pengantaran barang.
2. Sistem informasi geografis dapat digunakan untuk merepresentasikan lintasan terpendek.
3. Urutan inputan tidak mempengaruhi pemilihan lintasan terpendek karena algoritma A* akan mencari lintasan yang optimum.
4. Algoritma A* dapat diterapkan untuk graph berbobot dan tidak berarah.
5. Akurasi dalam proses digitasi dan registrasi peta akan mempengaruhi akurasi pemilihan lintasan terpendek karena semakin akurat koordinat *node* maka semakin akurat pula hasil pemilihan lintasan terpendek yang ditampilkan.

5.2 Saran

Saran-saran yang dapat digunakan untuk pengembangan skripsi ini adalah sebagai berikut :

1. Diharapkan *node* dan *edge* yang tersedia lebih lengkap dan dinamis sehingga peta yang tersedia lebih menyerupai keadaaan lapangan yang sebenarnya dan total jarak yang dilalui lebih akurat.
2. Aplikasi dapat dilengkapi dengan menambah perhitungan kapastitas barang yang dapat dimuat dalam kenderaan pengantar barang.

3. Penambahan parameter dalam menentukan lintasan terpendek, seperti titik *traffic light*, jalan rusak, dan titik macet dapat diperhitungkan dalam aplikasi ini.
4. Aplikasi dapat dijalankan tidak hanya pada *desktop*, tetapi dapat pula dijalankan melalui internet, *handphone* atau *gadget* agar user lebih mudah mengakses aplikasi.